Practical Protection for Public Servers

Joe Spagnolo
NRNS Incorporated
4043 Carling Avenue
Suite 106
Ottawa, Ontario K2K 2A3
Canada

Joe.Spagnolo@nrns.ca

ABSTRACT

The protection of public servers presents a challenge due to their high level of exposure. We present a practical approach to protecting public servers based on experience within a defence research and development network. Although our defence-in-depth approach has proven effective in protecting public servers, we believe the protection posture can be further improved. We outline the areas in which these improvements can be made, and discuss areas such as logging, intrusion detection, event correlation and automated response that we have not yet fully addressed in practice.

INTRODUCTION

The research and development (R&D) agency of the Canadian Department of National Defence, Defence R&D Canada (DRDC), operates an unclassified network called the DREnet. The DREnet supports both R&D initiatives as well as some of the agency's business activities. The DREnet was the first Canadian network connected to the Internet's predecessor – the Arpanet. Although the network is only used to process unclassified information, the protection of the network has always been a top priority for the agency. Early versions of the proprietary DREnet router systems included a packet filtering capability based on Internet Protocol (IP) addresses. After an incident in 1993 in which files were copied covertly from a private DREnet server by Internet based hackers, it became evident that filtering above the network layer was required. The next evolutionary step was to filter well-known protocols by their port numbers since the source of an attack could not be predicted. DRDC adopted strict firewall-like filtering on its border routers in 1993 and then in 1997 deployed commercial firewalls, which were followed later by virtual private network (VPN) encryption and intrusion detection technology.

Although the DREnet would not be considered a large network in this day and age, the DREnet does offer all the challenges associated with the operation of an Internet-connected network. More specifically, the DREnet offers public services that are susceptible to network based attacks since conceivably the associated servers need to communicate directly with any Internet client regardless of the time of day or the physical location of the user. In addition to perimeter protection, DREnet public servers utilize several host-based protection mechanisms and are offered a number of network-based protection mechanisms to shield them from intruders.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks",
held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **01 NOV 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE

Practical Protection for Public Servers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NRNS Incorporated 4043 Carling Avenue Suite 106 Ottawa, Ontario K2K 2A3 Canada

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

See also ADM001845, Adaptive Defence in Unclassified Networks (La defense adaptative pour les reseaux non classifies)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**32**	

THE CHALLENGE

An organization that connects its unclassified internal network or "Intranet" to public networks such as the Internet exposes its internal systems to numerous threats and risks. Public servers present an even greater challenge since the user of the service initiates the communication at a time and from a location of his choosing. Since the information served by these public servers is unclassified and readily available to the public, the concern is not normally the information’s confidentiality but rather its availability and integrity. As such, the server requires adequate protection to minimize the risk of compromise and the server must be monitored to detect unauthorized and malicious activity.

A typical protection posture usually begins with perimeter protection at all external connection points, with firewalls as the systems of choice for protecting an organization's Intranet from unwanted intruders. Firewalls alone, however, do not provide sufficient protection for public servers since firewall policies can be circumvented and firewall inspection engines often do not detect malicious activity that occurs within what appears to be the normal course of the client/server dialogue. As such, the firewall will not prevent a server compromise if the server is executing vulnerable software accessible through permitted traffic flows.

Depending on the intended purpose of an unclassified network, its network operations centre might only be staffed during local business hours (8 hours per day / 5 days per week - 8/5 management), with no on-call support. Public servers however should provide service around the clock 365 days per year in order to satisfy client requests. This presents additional challenges since the public servers would then operate without supervision more than 75% of the time. An attack against a public server that occurs during silent hours would likely only be detected the following workday, which in the case of a holiday weekend could be several days later. If a public server is compromised, it can be used as a launch point to attack other public servers or even internal systems within the Intranet. Information theft is now a possibility since a compromised public server can be used to steal sensitive information from an internal system.

The challenge, then, is to create a defence-in-depth approach to protecting public servers from unauthorized access. Protection can be afforded by network security devices such as firewalls and filtering routers, by properly configured host operating systems, as well as by correctly configured and tested server software. A strategic combination of these protection mechanisms will mitigate the risk of a server compromise and increase the information’s availability and integrity. Monitoring and the ability to respond are equally as important as the protection mechanisms since it is imperative that an intrusion be detected and dealt with as soon as possible. If a compromise occurs during the silent hours, a system that permits the infrastructure to react by disabling the public server, containing the intruder, or performing some other type of predefined action would be beneficial.

Technology alone cannot protect a network against attack. The technology must be applied in a sensible fashion by knowledgeable and skilled personnel who are able to assess the latest threats, recognize attacks and adapt the protection posture in order to mitigate the risk of server compromise.

THE DE-MILITARIZED ZONE

Ideally, any public server that communicates directly with Internet clients should reside in an isolated enclave known as a De-Militarized Zone (DMZ). A DMZ provides isolation between the public network and the organization’s Intranet. A DMZ can be deployed in many configurations such as in the common architectures shown in Figure 1. An *Attached DMZ* is connected to a third network interface on the firewall, while an *In-Line DMZ* resides between two firewalls. An *Internal DMZ* is not located with the organization’s publicly facing connections but rather resides within the protected Intranet. Like the *Attached DMZ* and the *In-Line DMZ*, a firewall provides isolation between the *Internal DMZ* and the Intranet.

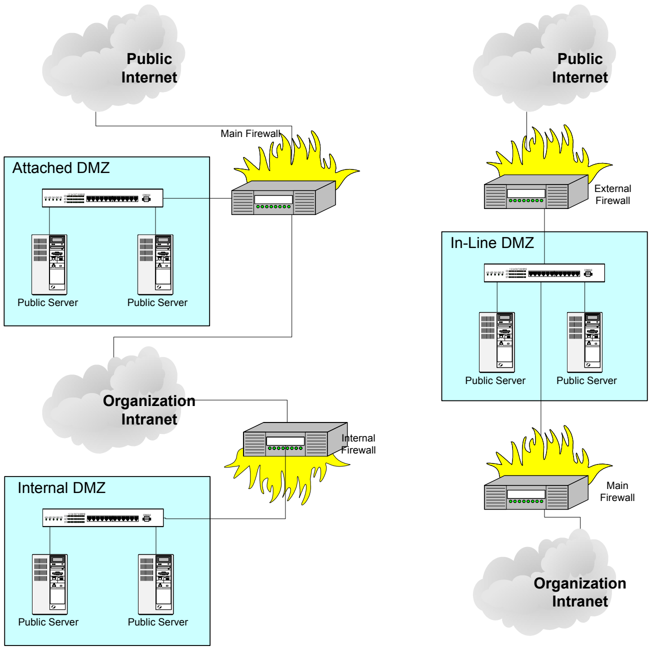


Figure 1 - DMZ Configurations

POLICY

A Security policy dictates the rules and regulations for establishing a public server; i.e. how public servers interact with public networks such as the Internet. A typical policy for the establishment of a public server may include the following statements:

- A public server must be established on dedicated hardware.
- A public server must reside within a DMZ segregated from the public network as well as the Intranet by a firewall.
- Only the communication absolutely required to provide the public service must be permitted by the firewall.
- A public server must execute software free of known vulnerabilities.
- All system and network services that are not absolutely required to provide the public service must be disabled on the public server.
- A public server must be scanned before it is deployed to ensure it is not executing software with known vulnerabilities.
- A public server must be periodically scanned to ensure it is not susceptible to newly discovered vulnerabilities.

- A public server must be monitored to detect attacks as well as compromises.
- A public server must be immediately disabled if it is suspected of compromise.

BASE SYSTEM CONFIGURATION AND VALIDATION

A most important aspect of a public server's security is its own configuration. If the server is comprised of properly configured software that is free of known vulnerabilities, the chance of successful compromise by an attacker is greatly reduced.

It is imperative that the underlying operating system be secured as much as possible. Relevant operating system patches must be applied to the system on a continual basis as made available by vendors. We do not recommend however that all operating system patches should be blindly applied. Some patches do not affect security, but may influence the operation of the application. As the system administrators, we require a good understanding of the system as well as the application in order to determine if patches are relevant or not.

Only the system and network services absolutely required to provide the intended service should remain enabled. The default configuration of both Unix and Windows based systems is likely to include insecure services. File and print sharing services, such as Network File System (NFS) on Unix systems and Microsoft Networking on Windows systems, must be completely disabled since these services are not typically required by public servers. Numerous other services also need to be assessed to determine whether they are required or not. If not required they should be disabled and, if possible, removed from the system. Respectable security conscious organizations have published numerous documents describing how to secure both Unix and Windows based systems [1] [2].

Of equal importance is the application software that implements the intended service. The server software must be free of known vulnerabilities and must be properly configured. For instance, a web server should not permit clients to list and navigate its file system. One must never assume that the default configuration is sensible. Instead, all configurable settings should be reviewed and set to achieve the most secure configuration possible while still meeting the defined business requirements.

Once the server configuration is complete, the public server system must be scanned to identify open ports, review the configuration, and uncover vulnerabilities. The scans must be repeated each time the server software is modified, when the server's configuration is altered, or when a new vulnerability is discovered. From time-to-time, scans must also be done to confirm the server's configuration and account for configuration drift.

PROTECTION MECHANISMS

We employ a defence-in-depth approach to heighten the protection offered to a DREnet public server. This section identifies a number of mechanisms for protecting public servers. While most of the mechanisms can be implemented on both Unix and Microsoft Windows based platforms, some are only available on Unix based platforms.

The sample security policy presented in this paper requires that the public server reside within a DMZ segregated from the public network as well as from the Intranet by a firewall, with the firewall providing the first line of defence. A firewall may also offer protection against certain denial of service attacks that are designed to deplete public server resources through the initiation of partially opened transport layer connections. A firewall can ensure that connections to public servers are fully established and that data-less connections are subsequently terminated.

Only traffic absolutely required to provide an intended service should be permitted to pass through the firewall. This not only includes traffic between Internet based clients and the public server, but may also include traffic between the public server and a back-end server such as a database server located within the Intranet. All other traffic to/from the public server should be denied and may be logged depending upon its potential and assessed impact. This is known as ingress and egress filtering.

Many firewalls on the market today are based on stateful inspection technology, which offers better performance than application proxy based firewalls. It can be argued however, that stateful inspection firewalls do not provide adequate protection since they do not fully interpret the application layer dialogue and therefore are not able to detect malicious activity at the application layer. Leading stateful inspection firewall vendors have recognized this deficiency and improved their products to monitor the application layer dialogue better and eliminate malicious intent. The latest Checkpoint firewall software with Application Intelligence [3] is currently being looked at to determine its effectiveness in discovering and eradicating malicious traffic. Application layer dialogue can also be monitored with in-line security devices from vendors such as E-Safe [4] and Radware [5] that can detect and purge malicious activity.

A network-based firewall cannot prevent a public server in the DMZ from attacking another public server that also resides in the same DMZ. Virtual local area network (VLAN) mechanisms can segregate public servers located within the same DMZ. Simple port based VLANs configured in a Local Area Network (LAN) switch can prevent communication between public servers that share a DMZ segment. Our configuration includes a distinct VLAN configured for each public server, with each VLAN consisting of only the public server and the firewall. This simple configuration forbids all communication between servers connected to the same LAN switch and therefore prevents the use of a compromised server to attack another server. As with all DREnet firewalls we maintain configuration control of all LAN switches used to implement DMZs regardless of their physical location. If VLANs are incorporated into the protection posture, the associated LAN switches must be configured in a secure manner to prevent tampering with the VLAN configuration.

Each public server should include a host-based firewall to control how it communicates on the network. We employ the packet filtering and firewall capabilities available on most Unix based systems to place tight controls on the server's communication. The host-based firewall policies, which effectively mimic the policies configured in the network-based firewall, protect the public server from attacks launched from other servers co-located within the same DMZ in the event that the segregation provided by the configured VLANs fails. The host-based firewall policies also protect the public server in the event an attacker is able to circumvent the main firewall. On one occasion, Internet intruders discovered a poorly configured Formmail.pl module on a DREnet web server and attempted to send unsolicited bulk email from the web server. The host-based firewall on the server blocked the outgoing Simple Mail Transfer Protocol (SMTP) connections since it was not traffic associated with the delivery of web services. It is interesting to note the effectiveness of the layered protection in this case: the same SMTP connections would have been blocked by the network-based firewall if the host-based firewall had not intervened.

Whenever possible, we do not permit processes that provide network services to execute with system level privileges. If a process with system level privileges is compromised, the attacker inherits the system level privileges, which provide unrestricted access to the system. Unfortunately, most operating systems do not permit non-privileged processes to bind to privileged transport ports [6] in the 1-1023 range, making privileged access necessary to run some applications. Some software supports the circumvention of this restriction by permitting the server process to accept incoming connection requests on privileged ports, but then passing the "open" socket to a child process that executes with restricted user level privileges. If a process with restricted user level privileges is compromised, the attacker only achieves restricted access to the system. To limit the damage that an intruder can inflict, we grant server processes read-only access to system files and directories as well as to server configuration and content files. For instance, the *httpd* process that provides public web services executes as a non-privileged user named *httpd.* If the *httpd* user is granted read-only access to the web content files, the intruder is not able to modify the content and deface the web site.

Most remotely exploitable vulnerabilities are associated with buffer overflow conditions introduced by careless programmers. Stack-smashing attacks [7] attempt to exploit buffer overflow vulnerabilities by corrupting the processor's execution stack and invoking the attacker supplied code. When supported by the processor (e.g. Sun Sparc), we enable an operating system feature that prevents execution of code on the processor stack. This operating system feature, although not 100% fool-proof [8], foils most buffer overflow related vulnerabilities since it causes the process to incur a segmentation violation before the process can be hijacked by the attacker. Other software based approaches, such as IBM's Stack-Smashing Protector (also known as ProPolice) [9], insert code at compile time that protects applications against stack-smashing attacks. IBM provides instructions on how to build complete Linux and FreeBSD systems with protection against stack-smashing, while OpenBSD now includes the protection directly within its software distribution. Commercial offerings that provide similar stack-smashing protection include the Immunix Secure Linux Operating System [10]. Microsoft is also incorporating code to guard against stack-smashing attacks in its compiler [11] [12] and hopefully stack-smashing protection will form part of future releases of Microsoft operating system software. Whether implemented in hardware or software, stack-smashing protection can not only prevent the exploitation of newly discovered buffer overflow vulnerabilities but it can also alert the system administrator that the server is under attack.

Executing the server process as a non-privileged user can limit the damage an intruder can inflict in the event the server process is compromised. Although the intruder’s inherited access rights may limit her actions, the intruder still possesses a complete view of the file system. Since a non-privileged local user can gain system level privileges through a locally exploitable vulnerability, the intruder’s view of the file system should be minimized. We create a virtual environment for the server process that appears like the real file system but in reality is simply a small subset of the file system. Unix systems include commands such as *chroot* or *jail* that implement a virtual sandbox environment that can be used to imprison the server process as well as the intruder if the server process is compromised. The use of a sandbox is especially attractive if the server process must execute with privileged rights. When the sandbox is created, only the files that are absolutely required to execute the server process are replicated in the sandbox environment. These typically include server binary and configuration files, a small number of required system libraries and a small number of required system configuration files. Consider an exploit for a buffer overflow condition that overflows the server’s buffer with code that attempts to execute a command shell (/bin/sh). If the /bin/sh executable file is not present in the sandbox environment, the exploit fails.

Figure 2 illustrates a simple sandbox environment. The Real File System contains system startup files such as */etc/rc.conf*, system binaries such as */bin/sh* and shared system libraries such as */usr/lib/libc.so* and */usr/lib/libm.so*. The Virtual Sandbox only contains the required system files, system binaries and system libraries in addition to the server binary */bin/server*.

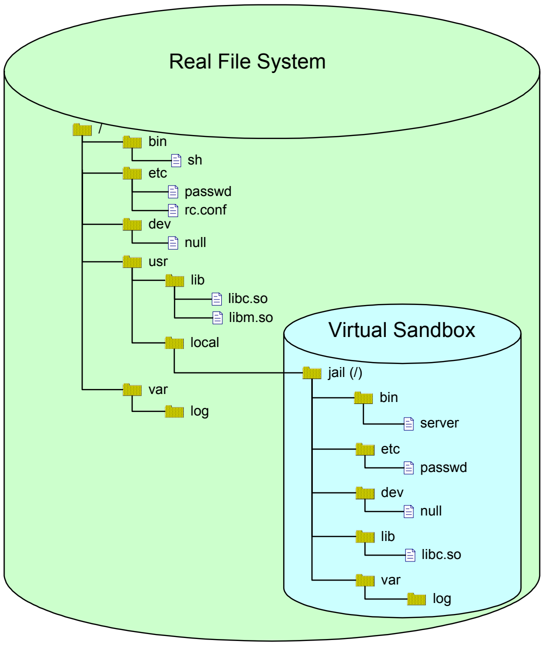


Figure 2 - The Virtual Sandbox

Imprisoning a server process in a sandbox environment mitigates the risk of compromise and reduces the damage the attacker can inflict if the server process is compromised. If the attacker inherits system level privileges and the attacker possesses detailed knowledge of the compromised system, it may still be possible for the attacker to alter the system configuration from within the sandbox environment. We enable kernel level security available on certain Unix variants that would restrict access to certain system operations regardless of the user's privilege level. Examples of system operations that would be restricted include:

- The ability to modify system files whose "immutable" flag is set.
- The ability to load and unload kernel modules.
- The ability to open raw disk devices and kernel memory devices for writing.
- The ability to alter packet filter and firewall rules.

Since the kernel security level cannot be lowered at run-time, the attacker must alter the contents of a system startup file and force the system to reboot. Recall however that an intruder trapped within a sandbox environment does not have access to the real system configuration files such as the system startup file.

REPLICATION OF CONTENT

The security mechanisms discussed in this paper provide protection against server compromise and restrict the attacker's activities in the event the server process is compromised. As an added precaution, a public server should not possess the master copy of the information it serves. If the public server possesses only a secondary copy of the content, the attacker is not able to alter the master copy of the information. We are contemplating the use of file/content synchronization software to eliminate the risk of corruption to the main copy of the content. For instance, a web server's content should be produced and maintained on a separate content server that resides within the Intranet. Ideally, the content should be pushed from the content server to the public server. In the event a compromise occurs and the integrity of the information served is in question, it will then be possible to restore altered information such as defaced web pages. The Microsoft Content Deployment Service (CDS) [13] facilitates replication for Microsoft web environments; the open source *rsync* software [14] can be used to synchronize web content in Unix web environments; while products such as the RepliWeb Deployment Suite (RDS) [15] may be better suited for heterogeneous environments.

The content server should initiate the replication process. The number of connections initiated by public servers to Intranet destinations must be kept at an absolute minimum. It is imperative that the public server confirms the identity of the initiator using some form of authentication. As discussed in the "Protection Mechanisms" section of this paper, the public server process must be granted read-only access to content files to prevent tampering by intruders. As a result, the replication service on the public server must have the ability to set the ownership and access rights on the content files accordingly.

MONITORING AND INTRUSION DETECTION

Intrusion Detection System (IDS) technology monitors network activity in hopes of identifying attacks and intrusions. Network based Intrusion Detection System (NIDS) sensors are deployed in strategic locations within the network infrastructure to monitor passively network traffic to detect attacks and intrusions. Host based IDS (HIDS) solutions are located on the server itself and monitor system activity.

A NIDS sensor should be deployed within each DMZ to monitor the network activity of public servers. Unlike a NIDS sensor assigned to monitor all traffic entering or leaving the Intranet, a DMZ NIDS sensor can be precisely tuned according to the services provided by the public servers. A NIDS specifically tailored to understand the server's intended use can recognize anomalies in the server's network interaction. For example, if the purpose of a public server is to provide web services, then it should only receive HTTP traffic on TCP port 80 and respond to initiated TCP sessions established via this port. That being the case, the NIDS should be configured to raise an alarm if it detects the public server attempting to communicate on ports other than TCP port 80. The presence of rogue traffic would likely be the result of either a faulty firewall configuration or an incorrect server configuration, or else intrusion attempts and compromises.

A HIDS checks for anomalies in system activity such as unauthorized processes and network services as well as altered system files. Some even check for kernel level instructions that appear malicious. Tripwire [17] in particular is very effective at establishing a system baseline for use in detecting modifications to system files. Once the HIDS establishes a baseline for system files, any attempt by an intruder to create or modify a file is detected and reported by the HIDS. If a service is implemented within a sandbox environment, the HIDS should employ finer-grain monitoring of the sandbox environment. An attacker who gains system level privileges on a public server may attempt to disable the HIDS before the HIDS detects his presence. If the intruder is imprisoned within a sandbox environment, the intruder is unable to access HIDS configuration and baseline files that reside outside the sandbox environment.

If our protection posture includes an automated content update or replication mechanism, we must synchronize its activities with that of the HIDS. Each time the replication mechanism updates the server content, the HIDS must re-establish its baseline. Otherwise, the HIDS will incorrectly interpret newly updated files as a server compromise. Ideally, the replication mechanism should stop the service while it updates the server content and only restart the service after the HIDS has re-established its baseline. This may be difficult to achieve since the HIDS and content replication mechanism usually are not available as a single integrated product or solution.

LOGGING AND CORRELATION

Ideally, all security devices and public servers should log all events to a common repository where event correlation processing can occur. This common log server should be configured in a secure manner using the same protection mechanisms offered to public servers. Intruders generally want to eliminate any evidence of their covert activities. When the logs are stored on a separate dedicated log server, the intruder is forced to compromise an additional system in order to conceal his presence. In any event, the integrity of the log data needs to be preserved and any tampering of the log data needs to be easily detected. *Syslog* replacements for Unix such as *Syslog-ng* [18] and *Modular Syslog* [19] employ cryptography to protect the integrity of the log data.

Hundreds of thousands of individual events may not seem suspicious when examined by the naked eye, but once the information is normalized and correlated, patterns may be discovered which cause attackers and intruders to be revealed. Currently, simple event correlation using locally developed scripts is often employed to identify the top sources and targets of suspicious activity. Although useful in a small environment, such locally developed scripts may not possess the sophistication to detect the activities of skilled attackers whose inconspicuous methods are hidden within hundreds of megabytes of log data. In addition, these scripts generally tend to scale rather poorly as the enterprise grows. Networks might benefit from utilizing enterprise security management products such as Intellitactics' Network Security Manager [21]. The event correlation and data mining capabilities of such products can be used to discover and analyze attack patterns across the entire enterprise. Unfortunately, enterprise class products typically demand an enterprise level price tag, which may not be easily justifiable in a lot of environments. Organizations should not underestimate the effort and associated cost required to deploy an enterprise security management solution. Although the organization should achieve immediate benefit from an enterprise security management solution, the software must be specifically tailored to reflect the local network environment. Event correlation in a network context is still an immature discipline and the subject of continued research.

STANDBY SERVERS

Once a server is compromised, it must be taken offline and carefully examined to determine the cause of the breach. This process can take hours, if not days, resulting in denial of service. We could restore the server from a recent backup, but the restored files likely would contain the same vulnerability that the intruder previously exploited to gain unauthorized access to the server.

If a prolonged service outage for a public service is not acceptable, a standby server built on a different platform could be activated to assume the duties of the compromised server. Consider the scenario where we establish public web servers on both a Microsoft Internet Information Server (IIS) platform and an Apache server on a Unix platform with both servers serving identical content. The primary public server actively satisfies client requests, while the standby server remains offline – either disconnected from the network or connected to a disabled LAN switch port. If we suspect the primary public server of compromise, we would disconnect it from the network immediately and we would activate the standby server to assume its duties. Of course, the standby server's content may not be up-to-date initially and would have to be synchronized with that of the content server. The compromised server could then be properly examined to determine the cause of the breach and subsequently rebuilt and fully patched so that it may resume its duties.

To date, the availability requirements for DREnet servers have not warranted the cost associated with the establishment and maintenance of separate standby servers.

AUTOMATED RESPONSE

For networks that are afforded only 8/5 management, it might be possible for an intruder to control a compromised server for several days before the breach is discovered. Since, in general, a compromised server should be disabled as soon as possible, we require an automated response mechanism that can detect a server compromise and disable the server during silent hours. Monitoring and intrusion detection technologies may detect when a public server is under attack or a public server is compromised. The response technique could be as simple as reconfiguring the DMZ LAN switch to disable the port that interconnects the comprised server to the DMZ upon receipt of a high severity alert. If a standby server is available, the same response technique could be used to activate the standby server so it can immediately assume the duties of the compromised server.

When designing automated response procedures, the response technique is usually not particularly difficult to implement. Rather the difficulty lies in deciding what should trigger an automated response. Relying on NIDS alerts alone may produce false positives and cause erratic behaviour from the automated response system. A combination of critical NIDS and HIDS alerts may provide sufficient evidence that a server is compromised, especially if the HIDS reports unauthorized processes, altered files or newly created files. As previously discussed in the "Logging and Correlation" section, events from all security devices and public servers should be logged to a common repository where the enterprise security management system can perform event correlation. As a result, the enterprise security management system is best equipped to determine if an automated response should be initiated. Although enterprise security management products can be configured to "react" based on a predefined set of conditions, they require high-quality information in order to identify effectively a successful server compromise. Without a high-degree of certainty, the result of the automated response system could be an unexpected self-imposed denial of service.

SUMMARY

Public servers present a security challenge since the user of the service initiates the communication at a time and from a location of his choosing. Well managed networks currently employ a defence in-depth approach to protect public servers from unauthorized access and this paper outlines additional measures that could be employed, particularly with respect to logging, intrusion detection and automated response. Figure 3 illustrates the target architecture for protecting and monitoring public servers on such a network. Its key features include:

Protection Mechanisms

1. A public server is established on dedicated hardware. It does not share a system with other services, especially services considered important to the organization’s day-to-day operations. The fewer services running on a system the easier it is to manage from an operational and security standpoint.

2. The underlying operating system is secured as much as possible. All security related operating system patches are applied to the system and only the system and network services absolutely required to provide the intended service remain enabled.

3. Before deployment, a public server system is scanned to identify open ports and to uncover any known vulnerabilities. The scans are repeated each time the server software is modified, when the server’s configuration is altered, and when new vulnerabilities are discovered. Scheduled scans are also done from time-to-time to confirm the server’s configuration.

4. Each public server, regardless of its physical location, resides within a DMZ segregated by a network-based firewall.

5. Only traffic absolutely required to provide the public service is permitted by the network-based firewall. All other traffic is dropped and depending upon severity is logged.

6. VLANs provide isolation between public servers that share the same DMZ. The DMZ switch prevents one public server from communicating with another public server.

7. A host-based firewall on the public server provides further protection from servers that share the same DMZ as well as protection from all other sources.

8. Public servers include mechanisms to guard against stack-smashing attacks.

9. Public server processes do not execute with system level privileges in order to minimize the damage that an intruder can inflict in the event the server is compromised.

10. The public server processes do not possess the ability to alter configuration and content files. A public server process is only granted read-only access to these files.

11. Whenever practical, public server processes reside in a sandbox environment to complicate attacker attempts to compromise the server and to minimize the damage that an intruder can inflict in the event the server is compromised.

12. The public servers employ kernel level security to protect critical system resources.

13. The public servers only possess a copy of the content, which is pushed to the public server from a content server that resides in the Intranet.

Monitoring

14. NIDS are configured to monitor the network activity of the public server within the DMZ. The NIDS configuration is specifically tailored to understand the server's normal network behaviour so it can recognize anomalies in the server's network interactions

15. A HIDS is active on each public server to monitor system activity such as unauthorized processes and network services as well as altered system files.

16. Events from all security devices and public servers should be logged to a common repository where event correlation can take place.

Standby Servers

17. To meet high-availability requirements, a standby server built on a platform different from the primary server is deployed to assume the primary server's duties in the event the primary server is compromised.

Automated Response

18. Automated response procedures are employed to disable a compromised public server and to activate the standby server if available. The enterprise security management system is best equipped to determine if an automated response should be initiated.

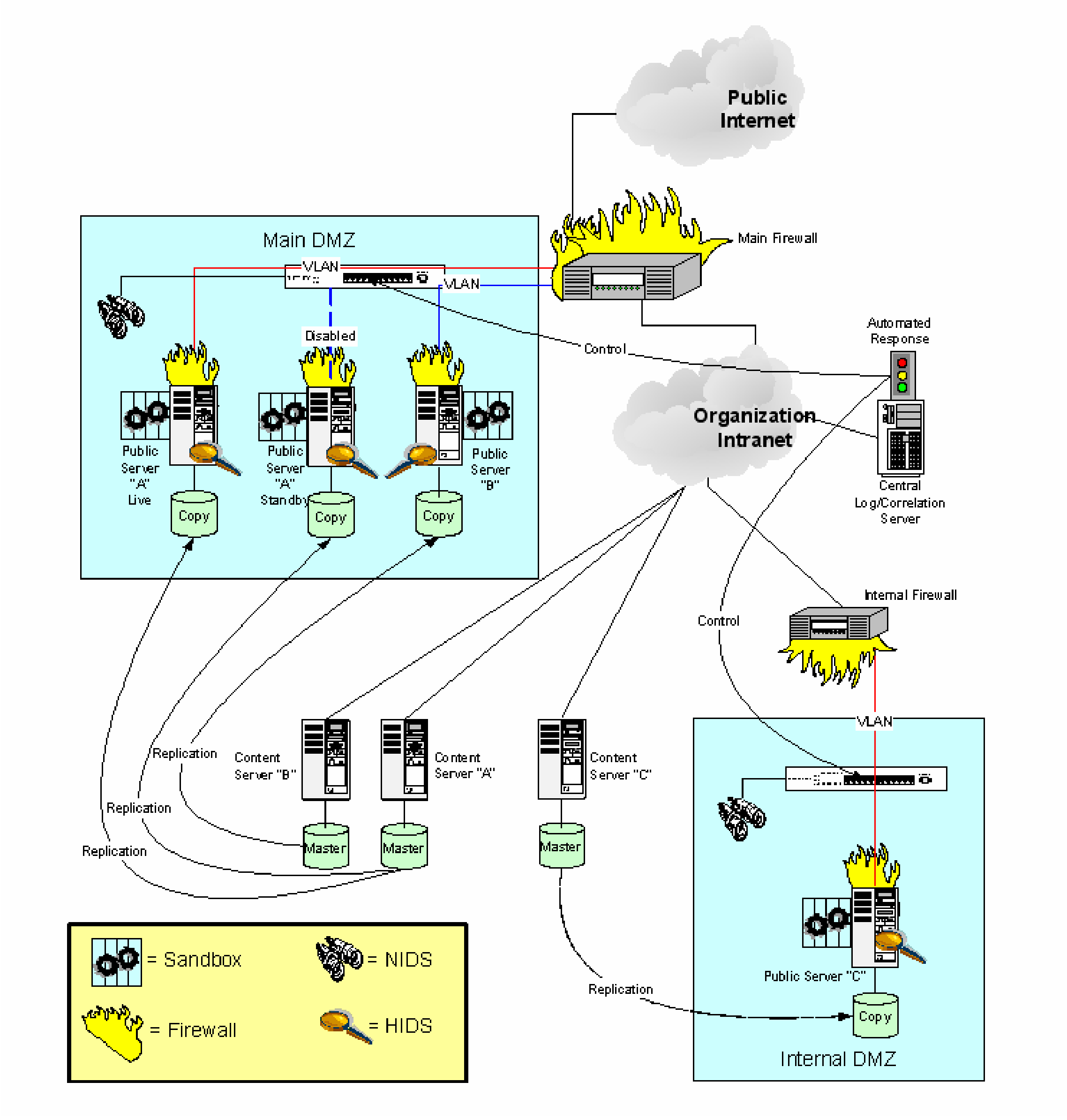


Figure 3 - Target Architecture

CONCLUSION

Much experience has been amassed within the DREnet in offering protection to public servers. An effective protection posture must start with the secure configuration and subsequent validation of the base system. If only the absolute minimum required system and network services are available on the system and the associated software is free of known vulnerabilities, the likelihood of a successful compromise is greatly reduced. When layered protection mechanisms are also implemented, the security of the public server is further enhanced. Most of the protection mechanisms described within this paper are relatively easy to implement. The strength of the resulting protection posture is directly attributable to the manner and depth in which the security mechanisms are applied.

It is difficult to determine how much protection is sufficient without the ability to monitor public servers. Our current monitoring and intrusion detection capabilities could still be strengthened since they cannot easily detect compromises and do not provide good visibility into the health and operation of the server. One should deploy NIDS sensors to monitor DMZ segments and deploy HIDS on each public server. A shortcoming of normal monitoring is that log information is collected and stored in numerous locations. Individual scripts can be used to achieve basic event correlation, but sophisticated event correlation requires the deployment of enterprise security management products. However, commercial enterprise security management solutions tend to be quite expensive and this cost must be weighed against their expected benefits.

An automated response system can disable a server if it believes the server to be compromised. A good automated response system is particularly important in unclassified environments where the servers can operate in unmanned environments up to 75% of the time. We believe we can develop the response technique without much difficulty by reconfiguring a DMZ LAN switch to disable the port associated with a compromised server. The challenge is in deciding what should trigger an automated response – a task best suited for the enterprise security management system since it possesses a broad view of the entire network. In the absence of an enterprise security management solution, however, the HIDS alone may provide sufficient evidence of a successful compromise since it possesses a detailed view of the server itself. The design and implementation of an automated response system requires a great deal of thought since it could result in an unexpected self-imposed denial of service.

We believe that a layered protection posture is an absolute requirement for protecting public servers. Although the DREnet only implements a subset of the target architecture presented in this paper, we already struggle to ensure configuration consistency across the various heterogeneous network devices, security devices and server systems. When a new service is deployed or an existing service is altered or retired, numerous devices must be independently reconfigured using the device's own management tool or interface. Without configuration tool support, this process can be error prone and could result in a weakened protection posture or improper server operation. This issue can be addressed by adopting a single vendor homogeneous solution or by the availability of a configuration audit tool that is able to operate with network and security products from different vendors.

ACKNOWLEDGEMENTS

This paper is based in large part on experiences gained and lessons learned through managing the DREnet over a number of years on behalf of Defence R&D Canada. The author would like to thank Vincent Taylor, Joanne Treurniet and Steve Zeber of Defence R&D Canada as well as Christopher James French of NRNS Incorporated for their advice and support during the writing of this paper.

REFERENCES

[1] SANS School Store, Step-by-Step Guides, https://store.sans.org/store_category.php?category=stepxstep

[2] National Security Agency, Security Recommendation Guides, http://www.nsa.gov/snac/

[3] VPN-1/FireWall-1 Gateways , http://www.checkpoint.com/products/protect/firewall-1.html

[4] Anti-virus, Anti-spam, Content Filtering Security , http://www.esafe.com/esafe/default.asp

[5] Defence Pro, First Strike Security, http://www.radware.com/content/products/dp/Default.asp

[6] IANA Assigned Port Numbers, http://www.iana.org/assignments/port-numbers

[7] Smashing The Stack For Fun And Profit, http://www.insecure.org/stf/smashstack.txt

[8] Defeating Solaris/SPARC Non-Executable Stack Protection, http://www.dtmf.com.ar/texts/non-exec-stack-sol.html

[9] GCC Extension for Protecting Applications From Stack-Smashing Attacks, http://www.trl.ibm.com/projects/security/ssp/

[10] StackGuard, http://www.immunix.org/stackguard.html

[11] Overview of Windows XP SP2 Security Technologies, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwxp/html/securityinxpsp2.asp

[12] Compiler Security Checks In Depth, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vctchcompilersecuritychecksindepth.asp

[13] Synchronization Fundamentals, http://www.microsoft.com/technet/prodtechnol/acs/proddocs/ac2k/acjgma_sycover.asp

[14] Welcome to the rsync Web Pages, http://samba.anu.edu.au/rsync/index.html

[15] RepliWeb File Replication and Content Distribution Software, http://www.repliweb.com

[16] The Open Source Network Intrusion Detection System, http://www.snort.org

[17] Home of the Tripwire Open Source Project, http://www.tripwire.org

[18] Syslog-ng, http://www.balabit.com/products/syslog_ng/

[19] CORE WISDOM – Secure Logging, http://www1.corest.com/products/corewisdom/CW01.php

[20] SILICON DEFENSE SnortSnarf Snort Alert Browser, http://www.silicondefense.com/software/snortsnarf/

[21] Intellitactics, Inc. - Enterprise Security Management Software, http://www.intellitactics.com

GLOSSARY OF TERMS

DMZ (De-Militarized Zone)

A DMZ provides isolation between the public network and the organization’s Intranet.

Enterprise Security Management

Enterprise security management products include event correlation and data mining capabilities that facilitate the discovery and analysis of attack patterns across the entire enterprise.

HIDS (Host Intrusion Detection System)

A HIDS is co-located on the server itself and monitors system activity such as unauthorized processes and network services as well as altered system files.

Intranet

An internal controlled network that is typically protected from the public uncontrolled network (i.e. the Internet) with the use of a firewall.

NIDS (Network Intrusion Detection System)

NIDS sensors are deployed in strategic locations within the network infrastructure to monitor network traffic passively in order to detect attacks and intrusions.

Stack-Smashing Attack

Stack-smashing attacks attempt to exploit buffer overflow vulnerabilities by corrupting the processor's execution stack and invoking the attacker supplied code.

Virtual Sandbox

A Virtual Sandbox provides a controlled operating environment for public server processes and is used to imprison the server process as well as the intruder if the server process is compromised.

VLAN (Virtual Local Area Network)

VLANs provide link layer isolation on local area networks. Network nodes can only communicate with members of the same VLAN.

Practical Protection For Public Servers

Introduction

- What is a public server?
  - A system that provides a network based service to external users.
- Why should it be protected?
  - Loss of service.
  - Prevent its use as a “springboard” for further attacks.

The Challenge

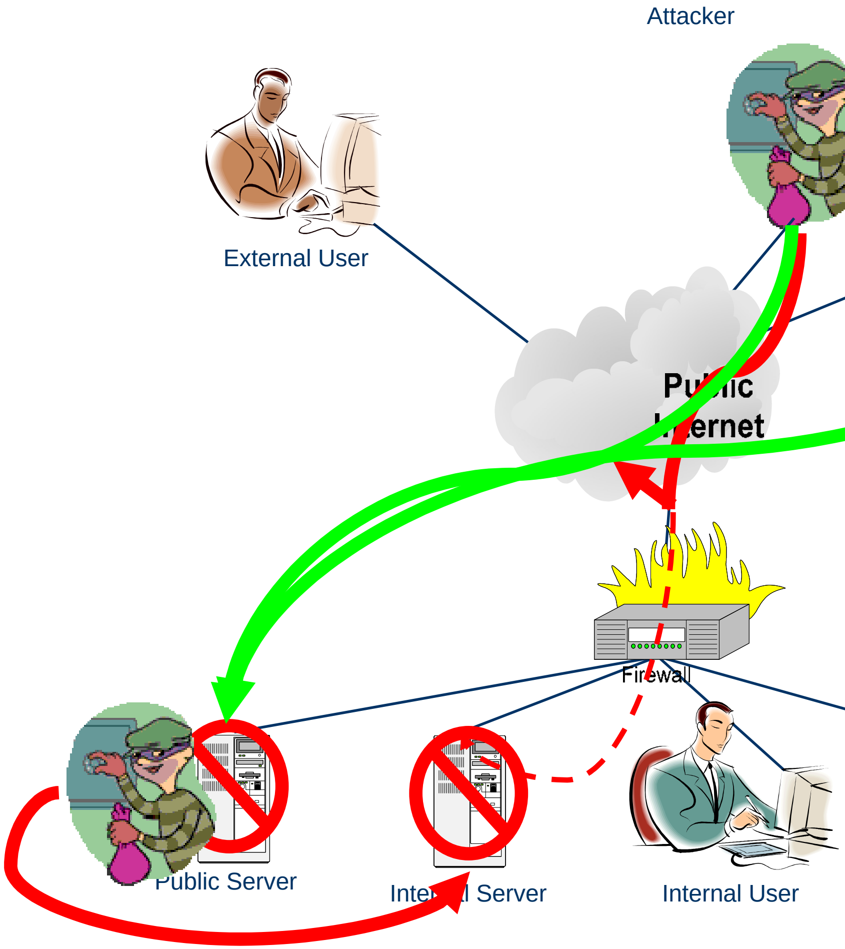
- Information availability and integrity.
- Staffing constraints.
- Monitoring and intrusion detection
- Automated response

Protection Policy

- A policy is a set of guidelines that governs the protection of a network.
- With respect to public servers, a policy may address:
  - Where the server is located.
  - How the server is established.
  - How the server is monitored.
  - Actions to take if the server is compromised.

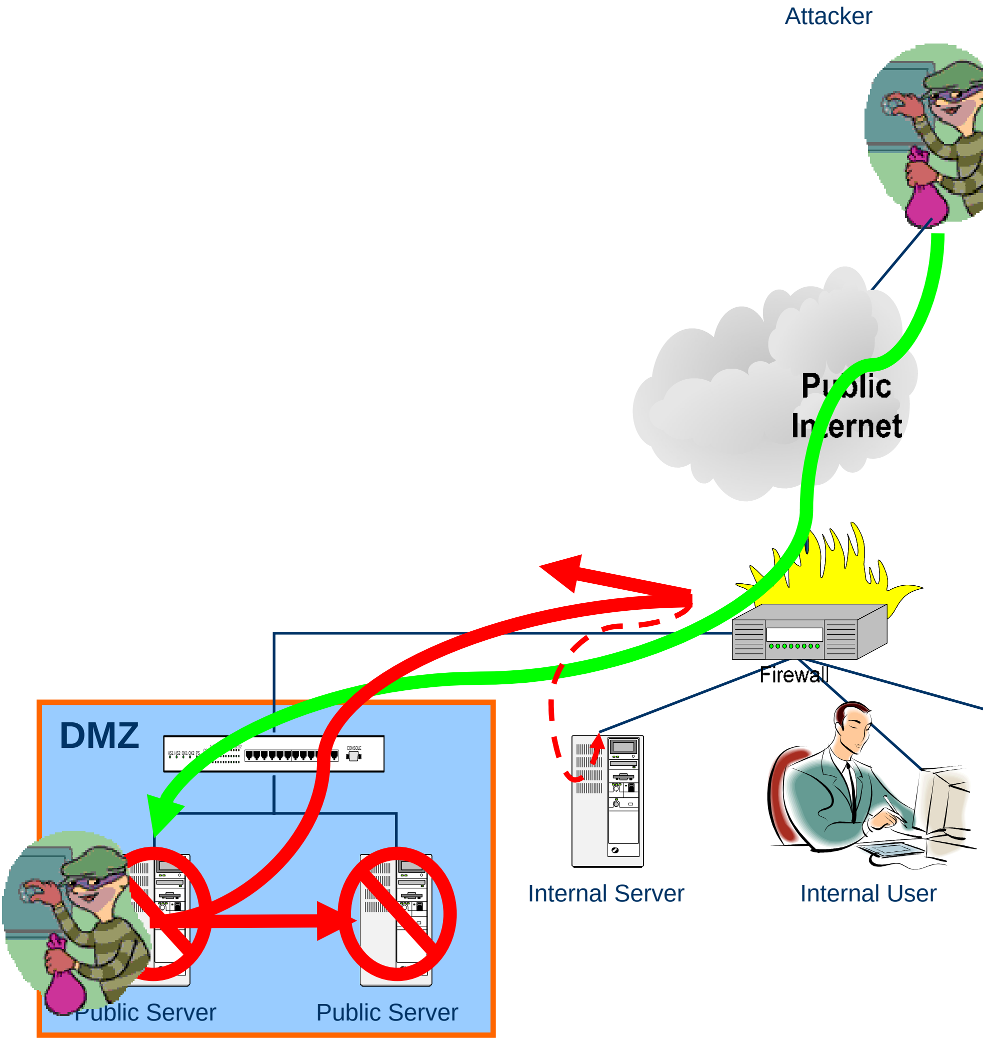
NRNS INFORMATION TECHNOLOGY PROFESSIONALS

Perimeter Based Security



Firewalls alone do not provide sufficient protection to Public Servers.

The Demilitarized Zone (DMZ)

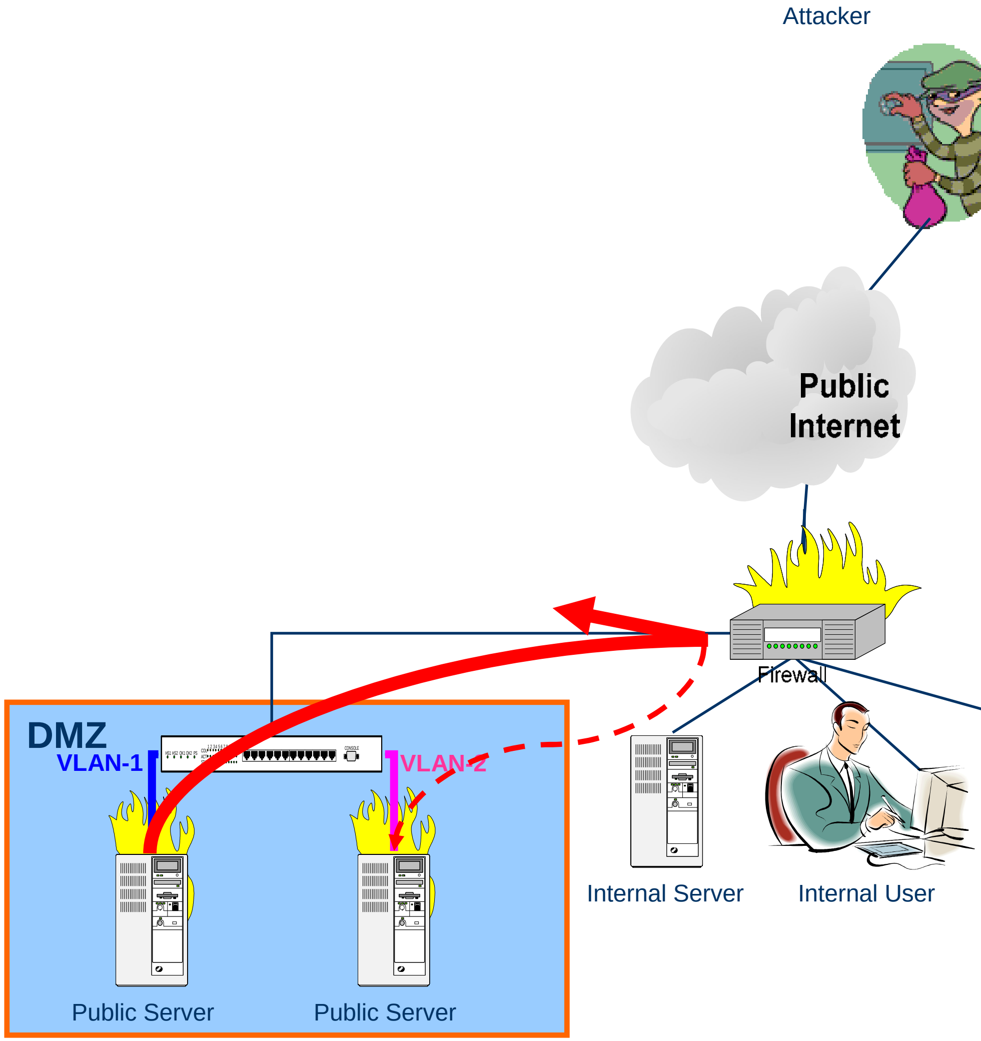


The DMZ itself does not offer any additional protection to the public server, but may prevent the intruder from infiltrating the internal network.

Start With a Strong Foundation

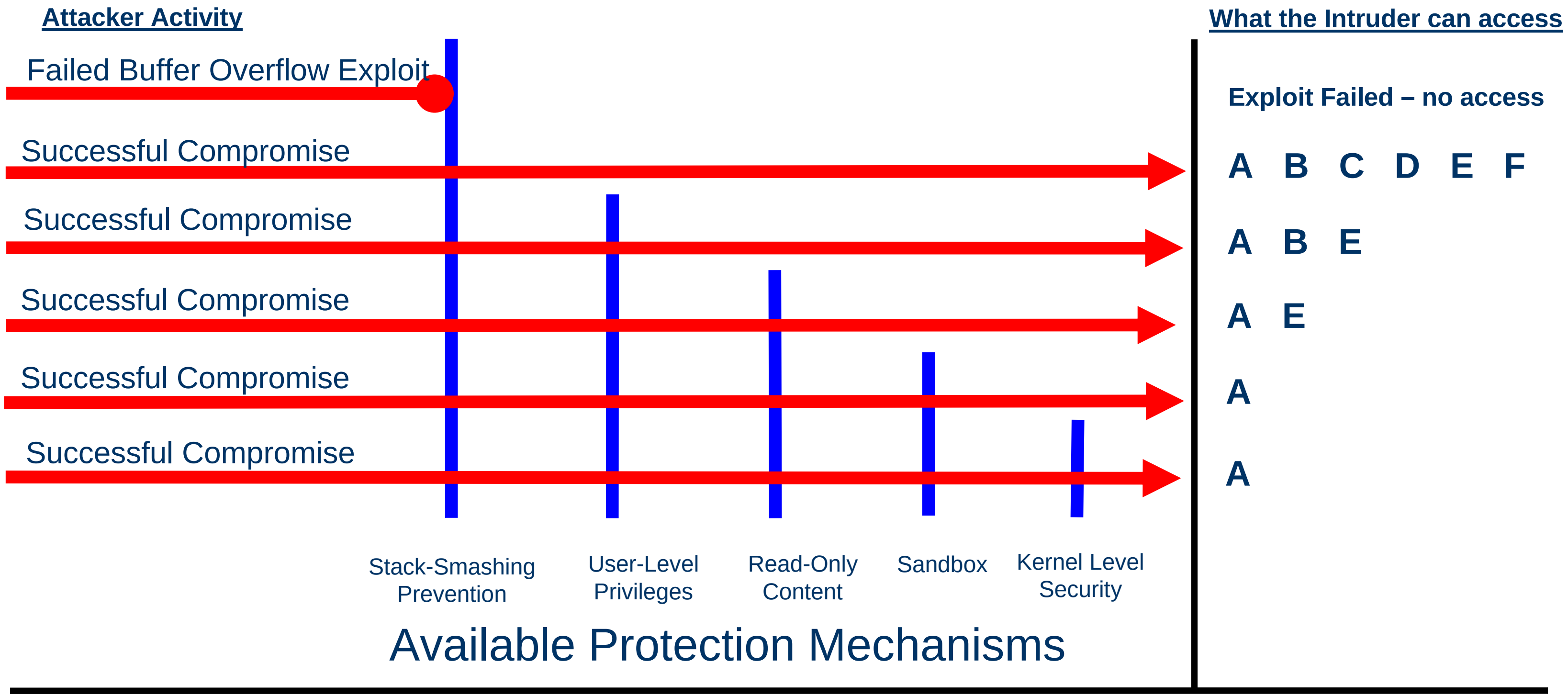
- Employ dedicated hardware.
- Apply relevant operating system patches.
- Disable or remove all non-essential services.
- Ensure server software is free of known vulnerabilities.
- Validate the server configuration.

Add Depth to Network Based Defences



Host-based firewalls and VLANs make it difficult for a compromised public server to attack another public server within the same DMZ.

Add Depth to Host-Based Defences



Resources Accessible by Intruder in the Event of Server Compromise

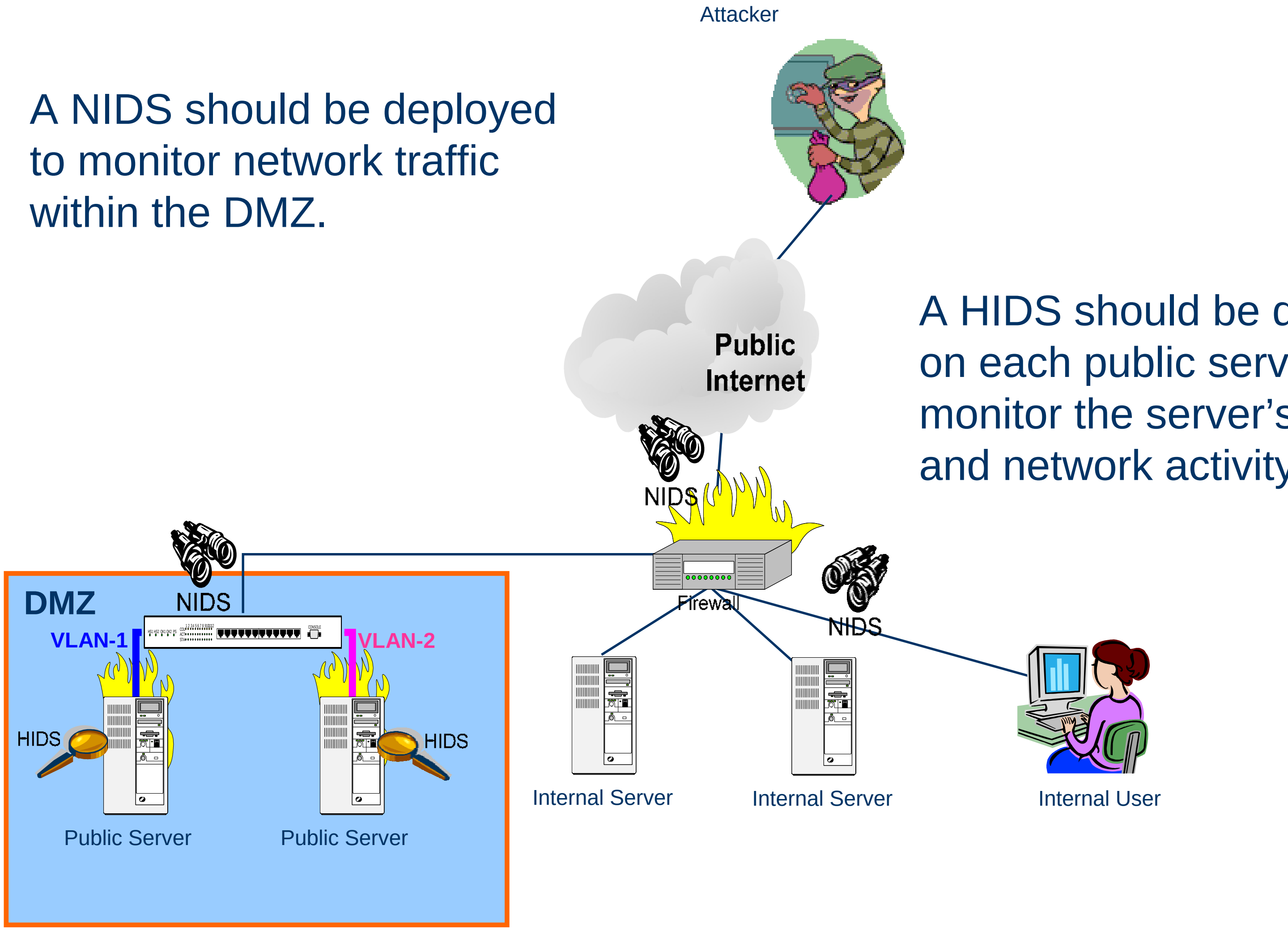
A = Server content files – read-only
B = Server content files – read/write
C = System database files

D = System privileged resources
E = Other executables
F = System restricted resources

Monitoring and Intrusion Detection

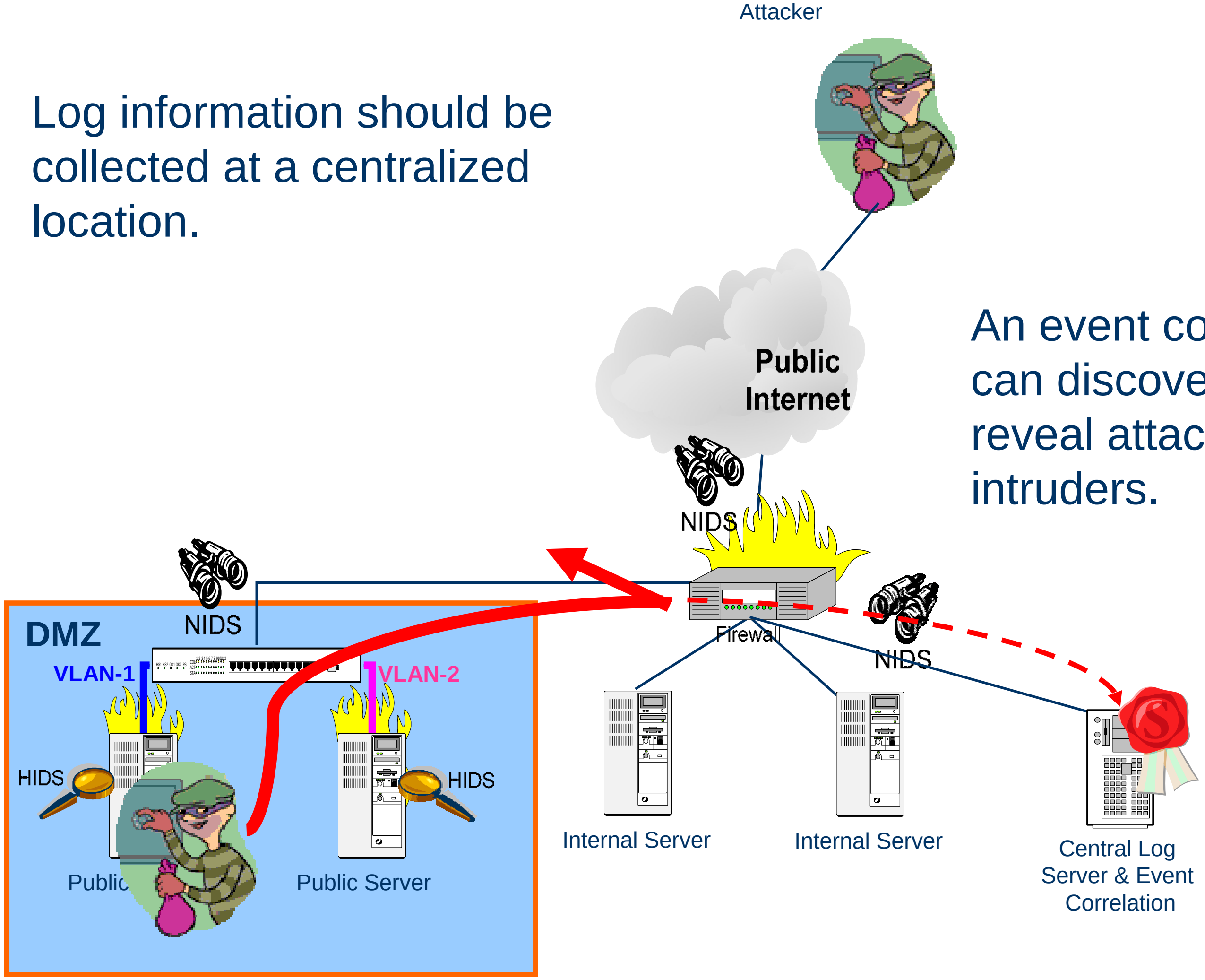
A NIDS should be deployed to monitor network traffic within the DMZ.

A HIDS should be deployed on each public server to monitor the server’s system and network activity.



Logging and Event Correlation

Log information should be collected at a centralized location.



An event correlation system can discover patterns that reveal attackers and intruders.

Standby Servers

- A compromised server must be taken offline immediately.
- How long will it take to examine a compromised server to determine the cause of the breach?
- How long will it take to rebuild a fully-patched server?
- How long can the public service remain unavailable?
- Is it feasible to pre-build a standby server based on a different platform?

Automated Response

- What if server compromise occurs during silent hours?
- Can an automated response mechanism detect the intruder and react?
- Disable the server and enable the standby server.
- What should trigger an automated response?
- How effective is the detection mechanism?
- The result may be an unexpected self-imposed denial of service.

Summary

- Need to start with a strong foundation.
- Employ layered protection mechanisms.
- Deploy both host-based and network-based IDS
- Collect log data in a central repository.
- Correlation of log data will assist in identifying attackers and intruders.
- A reliable trigger for an automated response system may be difficult to implement.

Where We Need to Focus our Attention

- Centralized log repository
- Integrity of log data
- Event correlation
- Automated response
- Consistency of configuration

Questions?